KoNLPy – 한글형태소 분석 모듈 : 파이썬

파이썬에서 문장열 분석으로 많은 인사이트를 얻어낼 수 있습니다. 영어의 경우는 단어만 구분하면 문자열 분석이 수월하지만, 한글의 경우는 형태소 단위로 분석해야 하기 때문에 까다롭습니다.

비영어권 언어들은 문자열을 분석하려면 인코딩과 디코딩 과정을 거쳐야 하고 문자에 맞는 분석방법을 사용해야 하기 때문에 어렵습니다. 특히 한글은 전 세계에서 분석이 가장 어려운 언어에 해당되기 때문에, 한글을 분석할 수 있다면 다른 나라의 언어들도 분석할 수 있습니다. 참고로 한글이 전 세계에서 13 번째로 많이 사용하는 언어라고 합니다

KoNLPy 패키지는 한국어 자연어 처리(NLP)를 위해 자주 쓰이는 파이썬 한국어 형태소 분석 패키지로 "코엔엘파이"라고 읽습니다. 이 패키지 안에 여러 종류의 한국어 형태소 분석기로 **Okt, Komoran, Kkma, Mecab** 이 있습니다.

[KoNLPy 설치 환경]

1) JAVA 1.7 이상 설치
2) JAVA_HOME Path 설정
3) JPype1 (>=0.5.7) 설치
'코엔엘파이'의 경우 JAVA 로 작성된 모듈을 로드하여야 하기 때문에 JPype1 0.5.7 이상이 설치되어야 합니다.

```
Pip3 install JPype1
```

4) KoNLPy 설치

```
pip install konlpy
```

[용어 정리]

형태소란?

형태소(形態素)는 '뜻을 가진 가장 작은 말의 단위'로 (출처: 국립국어원 표준국어대사전),
더 이상 나누게 되면 그 의미가 없어지는 것들을 말합니다.
예를 들어, '책가방' 은 '책', '가방' 이 두가지가 형태소라고 말할 수 있습니다.
가방에서 가/방으로 나누면 더이상 가방의 의미가 아니게 되니까 최소 단위가 '가방'이 된다고 볼 수 있습니다.

어간이란?

어간(語幹)은 '활용어가 활용할 때에 변하지 않는 부분'을 말합니다.
(출처: 국립국어원 표준국어대사전)
예를 들어, 동사 '보다'의 경우 '보았다(과거), 보니, 보고' 등으로 활용될 수 있는데
이들의 어간은 '보-'가 된다고 볼 수 있습니다.

어절이란?

어절(語節)은 '문장을 구성하고 있는 각각의 마디'를 말하며,
문장 성분의 최소 단위로서 띄어쓰기의 단위가 됩니다. (출처: 국립국어원 표준국어대사전)

'나는 자연어 처리 공부를 한다.'

위의 문장을 보면 '나는 / 자연어/ 처리/ 공부를/ 한다.'
이렇게 5 개로 나눌 수 있는데 이것이 각각 어절이 된다고 볼 수 있습니다.

품사란?

품사(品詞)는 '단어를 기능, 형태, 의미에 따라 나눈 갈래' 라고 합니다.
(출처: 국립국어원 표준국어대사전)
대표적으로 의미에 따라

1. 명사 : 사물의 이름을 나타내는 품사 ex) 사과, 달, 별
2. 대명사 : 사람이나 사물의 이름을 대신 나타내는 말 ex) 무엇, 이것, 저기
3. 수사 : 사물의 수량이나 순서를 나타내는 품사 ex) 하나, 둘, 셋, 일, 이, 삼
4. 조사 : 체언이나 부사, 어미 따위에 붙어 그 말과 다른 말과의 문법적 관계를 표시하거나 그 말의 뜻을 도와주는 품사 ex) 은, 는, 이, 가
5. 동사 : 사물의 동작이나 작용을 나타내는 품사 ex) 보다, 먹다, 자다, 일어나다

6. 형용사 : 사물의 성질이나 상태를 나타내는 품사 ex) 시원하다, 귀엽다, 기쁘다
7. 관형사 : 체언 앞에 놓여서, 그 체언의 내용을 자세히 꾸며 주는 품사 ex) 새, 옛, 윗, 뒷
8. 부사 : 용언 또는 다른 말 앞에 놓여 그 뜻을 분명하게 하는 품사 ex) 빨리, 다행히, 매우
9. 감탄사 : 말하는 이의 본능적인 놀람이나 느낌, 부름, 응답 따위를 나타내는 말의 부류 ex)아이고, 아차, 와

이렇게 9 가지로 분류할 수 있습니다. (출처: 국립국어원 표준국어대사전)

품사 태깅(tag)

형태소의 뜻과 문맥을 고려하여 그것에 마크업을 하는 일입니다.

예를 들어, 가방에 들어가신다 -> 가방/NNG + 에/JKM + 들어가/VV + 시/EPH + ㄴ다/EFN

한국어 품사 태그 비교표 링크 : https://docs.google.com/spreadsheets/d/1OGAjUvalBuX-oZvZ_-9tEfYD2gQe7hTGsgUpiiBSXI8/edit#gid=0

말뭉치(corpus)

Hannanum - KAIST 말뭉치를 이용해 생성된 사전

Kkma - 세종 말뭉치를 이용해 생성된 사전 (꼬꼬마)

Mecab - 세종 말뭉치로 만들어진 CSV형태의 사전

Komoran- Java로 쓰여진 오픈소스 한글 형태소 분석기

Twitter(Okt) - 오픈소스 한글 형태소 분석기

예제 소스 1 :

```
# Hannanum - KAIST 말뭉치를 이용해 생성된 사전
from konlpy.tag import Hannanum

hannanum = Hannanum()

# hannanum.analyze()  # 구(Phrase) 분석
# hannanum.morphs()  # 형태소 분석
# hannanum.nouns()  # 명사 분석
# hannanum.pos()  # 형태소 분석 태깅

# 사용예시
```

```
print('hannanum ----------------------------------')
print(hannanum.analyze(u'롯데마트의 흑마늘 양념 치킨이 논란이 되고 있다.'))
print(hannanum.morphs(u'롯데마트의 흑마늘 양념 치킨이 논란이 되고 있다.'))
print(hannanum.nouns(u'다람쥐 헌 쳇바퀴에 타고파'))
print(hannanum.pos(u'웃으면 더 행복합니다!'))

# Kkma - 세종 말뭉치를 이용해 생성된 사전 (꼬꼬마)
from konlpy.tag import Kkma

kkma = Kkma()

# kkma.morphs()  # 형태소 분석
# kkma.nouns()  # 명사 분석
# kkma.pos()  # 형태소 분석 태깅
# kkma.sentences()  # 문장 분석

# 사용예시
print('kkma ----------------------------------')
print(kkma.morphs(u'공부를 하면할수록 모르는게 많다는 것을 알게 됩니다.'))
print(kkma.nouns(u'대학에서 DB, 통계학, 이산수학 등을 배웠지만...'))
print(kkma.pos(u'다 까먹어버렸네요?ㅋㅋ'))
print(kkma.sentences(u'그래도 계속 공부합니다. 재밌으니까!'))

# Komoran- Java 로 쓰여진 오픈소스 한글 형태소 분석기
from konlpy.tag import Komoran

komoran = Komoran()

# komoran.morphs()  # 형태소 분석
# komoran.nouns()  # 명사 분석
# komoran.pos()  # 형태소 분석 태깅

# 사용예시
print('komoran ----------------------------------')
print(komoran.morphs(u'우왕 코모란도 오픈소스가 되었어요'))
print(komoran.nouns(u'오픈소스에 관심 많은 멋진 개발자님들!'))
print(komoran.pos(u'혹시 바람과 함께 사라지다 봤어?'))
```

```
# Twitter(Okt) - 오픈소스 한글 형태소 분석기
from konlpy.tag import Okt

okt = Okt()

# okt.morphs()  # 형태소 분석
# okt.nouns()  # 명사 분석
# okt.phrases()  # 구(Phrase) 분석
# okt.pos()  # 형태소 분석 태깅

# 사용예시
print(okt.morphs(u'단독입찰보다 복수입찰의 경우'))
print(okt.nouns(u'유일하게 항공기 체계 종합개발 경험을 갖고 있는 KAI 는'))
print(okt.phrases(u'날카로운 분석과 신뢰감 있는 진행으로'))
print(okt.pos(u'이것도 되나욬ㅋㅋ'))


# ex2) Konlpy.utils
# 클래스에서 유용한 메서드
# konlpy.utils.concordance(찾을 단어, 전체문장, show = False)
# concordance 단어색인(해당 단어가 쓰인 부분(index) 찾음)
# show = True 인 경우 단어 색인과 함께 포함된 문장 반환
# konlpy.utils.pprint(obj)  # 유니코드 문자 출력
# konlpy.utils.read_json(filename, encoding=u'utf-8')  # json 파일 읽음
# konlpy.utils.read_txt(filename, encoding=u'utf-8')  # txt 파일 읽음
# konlpy.utils.csvread(f, encoding=u'utf-8')  # csv 파일 읽음
# konlpy.utils.csvwrite(data, f)  # csv 파일로 쓰기 가능
# konlpy.utils.hex2char(h)  # 16 진수 문자를 유니코드 문자로 변환
```

예제 소스 2 :

```
####한국어 형태소 분석기####
# 각 모듈 비교
from konlpy.tag import Kkma, Komoran, Okt, Hannanum #Mecab 은 윈도우에서 작동 불가능
okt = Okt()
kkma = Kkma()
komoran = Komoran()
hannanum = Hannanum()
```

```
text = '나랏말이 중국과 달라 한자와 서로 통하지 아니하므로, ₩
    우매한 백성들이 말하고 싶은 것이 있어도 마침내 제 뜻을 잘 표현하지 못하는 사람이 많다.₩
    내 이를 딱하게 여기어 새로 스물여덟 자를 만들었으니, ₩
    사람들로 하여금 쉬 익히어 날마다 쓰는 데 편하게 할 뿐이다.'

#### .morphs()함수: 텍스트를 형태소 단위로 나누어준다.####
print("[Kkma morphs 함수]")
print(kkma.morphs(text))
print("[Okt 함수]")
print(okt.morphs(text))
print("[Komoran 함수]")
print(komoran.morphs(text))
print("[Hannanum 함수]")
print(hannanum.morphs(text))

##### stem: 각 단어에서 어간 추출 #####
print("[Okt 함수: stem 사용하여 어간 추출]")
print(okt.morphs(text, stem= True))

#### .nouns()함수: 명사를 추출 ####
print("[Kkma nouns 함수]")
print(kkma.nouns(text))
print("[OKt nouns 함수]")
print(okt.nouns(text))
print("[Komoran nouns 함수]")
print(komoran.nouns(text))
print("[Hannanum nouns 함수]")
print(hannanum.nouns(text))

#### .phrases()함수: 어절 추출 ####
print("[Okt phrases 함수]")
print(okt.phrases(text))

#### .pos()함수: 품사 태깅 ####
print("[Kkma pos 함수]")
print(kkma.pos(text)) #join=True 는 형태소와 품사를 붙여서 리스트화
```

```
print("[Okt pos 함수]")
print(okt.pos(text))
print("[Komoran pos 함수]")
print(komoran.pos(text))
print("[Hannanum pos 함수]")
print(hannanum.pos(text))
```

예제 소스 3 :

```
# konlpy 모듈에서 Twitter 클래스 사용
from konlpy.tag import Twitter
import tweepy
import jpype

# norm 형태소를 깔끔하게 만들어 주고, 불필요한 데이터를 지움
# stem 형태소의 원형을 찾아서 반환해 줌
twitter = Twitter()
text ="게임을 통해 디지털 트렌스포메이션의 자본적 수혜자가 일반 대중으로 확산되고 있고, 소비자에 머물렀던 일반 대중이 디지털 생산자로 변화하고 있다며 일부 인플루언서에  그치지 않고 일반 대중으로 확산될 것이며 디지털 컨텐츠는 대중이 접근하기 쉬운 아르바이트, 투잡이 투잡이 될 것이다."
malist = twitter.pos(text, norm=True, stem=True)
print(malist)
```

예제 소스 4 :

```
# 명사 중에 가장 많이 등장하는 단어 찾기
import codecs
import csv
from konlpy.tag import Twitter

twitter = Twitter()
word_dic = {}  # dictionary
lines = []  # list
```

```
# csv 파일에서 데이터를 읽어온다. 다른 파일로 변경 사용할 것.
with open('koreantest.csv', 'r') as raws:
    reader = csv.reader(raws)
    for raw in reader:
        lines.append(raw)

for line in lines:
    malist = twitter.pos('', join(line))
    print(malist)

    # 명사들을 수집해서 반봅되는 명사 count 를 진행한다.
    for word in malist:
        if word[1] == "Noun":
            if not(word[0] in word_dic):
                word_dic[word[0]] = 0
            word_dic[word[0]] += 1
keys = sorted(word_dic.items(), key=lambda x:x[1], reverse=True)

# 50개까지 결과값을 출력
for word, count in keys[:50]:
    print("{0}({1})".format(word, count), end="")
```

에제 소스 5 :

```
# 네이버 영화 리뷰 데이터의 10000개 문장에 대해 각 분석기의 소요 시간 비교
import time
from tqdm import tqdm

def tagger_time(tagger, texts):
  time_sum = 0

  for sentence in tqdm(texts):
    t1 = time.time()
    try:
      tagger.morphs(sentence)
```

```
        except:
            pass
        t2 = time.time()

        time_sum += (t2 - t1)

    return time_sum

# document : 네이버 영화 리뷰 데이터 (별도로 준비할 것)
texts = train['document'][:10000]
time_list = []

for tagger in [kkm, kom, okt, mec]:
    time_list.append(tagger_time(tagger, texts))

# 비교 분석 결과 시각화 처리
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('whitegrid')

tagger = ['Kkma', 'Komoran', 'Okt', 'Mecab']

plt.figure(figsize=(10,8))
plt.bar(tagger, time_list, color=(0.4,0.7,0.5))
plt.title('Learning Time for 10000 reviews', fontsize=17)
plt.xticks(fontsize=14)
plt.ylabel('total seconds')
```

[코드 단계별 분석]

from konlpy.tag import Okt txt = "아버지가방에들어가신다."	
# 인스턴스 생성 **# 인스턴스 : 클래스를 사용하기 위해 메모리 구조를 만든다.** okt = Okt()	
# 형태소 분석 Print(okt.pos(txt)) print(okt.pos("이것도 되나욬ㅋㅋㅋㅋ"))	[('아버지', 'Noun'), ('가방', 'Noun'), ('에', 'Josa'), ('들어가신다', 'Verb'), ('.', 'Punctuation')] [('이', 'Determiner'), ('것', 'Noun'), ('도', 'Josa'), ('되나욬', 'Noun'), ('ㅋㅋㅋㅋ', 'KoreanParticle')]
# norm = True : 품사 태깅(기본값 False) Print(okt.pos("이것도 되나욬ㅋㅋㅋㅋ",norm = True))	[('이', 'Determiner'), ('것', 'Noun'), ('도', 'Josa'), ('되나요', 'Verb'), ('ㅋㅋㅋ', 'KoreanParticle')]
# stem=True : 원형 글자로 바꿔준다. (기본값 False) Print(okt.pos("이것도 되나욬ㅋㅋㅋㅋ",norm = True,stem=True))	[('이', 'Determiner'), ('것', 'Noun'), ('도', 'Josa'), ('되다', 'Verb'), ('ㅋㅋㅋ', 'KoreanParticle')]
# 텍스트를 형태소 단위로 나눈다. Print(okt.morphs(txt))	['아버지', '가방', '에', '들어가신다
# 텍스트에서 명사만 추출 Print(okt.nouns(txt))	['아버지', '가방']

# 텍스트에 어절을 추출 Print(okt.phrases(txt))	['아버지가방', '아버지', '가방']

from konlpy.tag import Kkma	
# 인스턴스 생성 kkma = Kkma()	
# 형태소 분석 Print(kkma.pos(txt)) Print(kkma.pos("이것도 되나욬ㅋㅋㅋㅋ"))	[('아버지', 'NNG'), ('가방', 'NNG'), ('에', 'JKM'), ('들어가', 'VV'), ('시', 'EPH'), ('ㄴ다', 'EFN'), ('.', 'SF')] [('이것', 'NP'), ('도', 'JX'), ('되', 'VV'), ('나', 'ECE'), ('욬', 'UN'), ('ㅋㅋㅋㅋ', 'EMO')]
# 텍스트를 형태소 단위로 나눈다. Print(kkma.morphs(txt))	['아버지', '가방', '에', '들어가', '시
# 텍스트에서 명사만 추출 Print(kkma.nouns(txt))	['아버지', '아버지가방', '가방']
# 텍스트에서 문장을 분석 Print(kkma.sentences(txt))	['아버지가방에 들어가신다.']

[말뭉치 코드 분석]

```
from konlpy.corpus import kolaw
from konlpy.tag import Okt
import nltk
# 애국가.txt
```

```
# 말뭉치로 텍스트 읽어들이기
# 오류가 나면 텍스트 파일의 유니코드를 UTF-8
로 바꿔야함# C:₩사용자 경로₩anaconda3₩Lib₩site-
packages₩konlpy₩data₩corpus₩kolaw 경로에 있는 텍스트 파일만 읽어들일 수 있다.

doc_ko = kolaw.open("애국가.txt").read()

print(type(doc_ko))
print(doc_ko)
```

```
동해물과 백두산이 마르고 닳도록
하느님이 보우하사 우리나라만세
무궁화 삼천리 화려강산 대한사람 대한으로 길이 보전하세

남산위에 저 소나무 철갑을 두른듯
바람서리 불변함은 우리기상 일세
무궁화 삼천리 화려강산 대한사람 대한으로 길이보전하세

가을하늘 공활한데 높고 구름없이
밝은달은 우리가슴 일편단심일세
무궁화 삼천리 화려강산 대한사람 대한으로 길이보전하세

이 기상과 이 맘으로 충성을 다하여
괴로우나 즐거우나 나라사랑하세
무궁화 삼천리 화려강산 대한사람 대한으로 길이보전하세
```

```
# 말뭉치에서 단어 추출

okt = Okt()
token_ko = okt.nouns(doc_ko)
print(token_ko)
```

```
['해물',
 '백두산',
 '마르고',
 '하느님',
 '보우',
 '하사',
 '우리나라',
 '만세',
 '무궁화',
 '삼천리',
 '화려',
 '강산',
```

```
# 다시 단어들을 문장으로 만들기
ko = nltk.Text(token_ko)
print(ko)
```

```
<Text: 해물 백두산 마르고 하느님 보우 하사 우리나라 만세...>
```

```
# 중복되는 단어 제거

print(len(ko.tokens))
print(len(set(ko.tokens)))
```

```
len(ko.tokens)
```

```
71
```

```
len(set(ko.tokens))
```

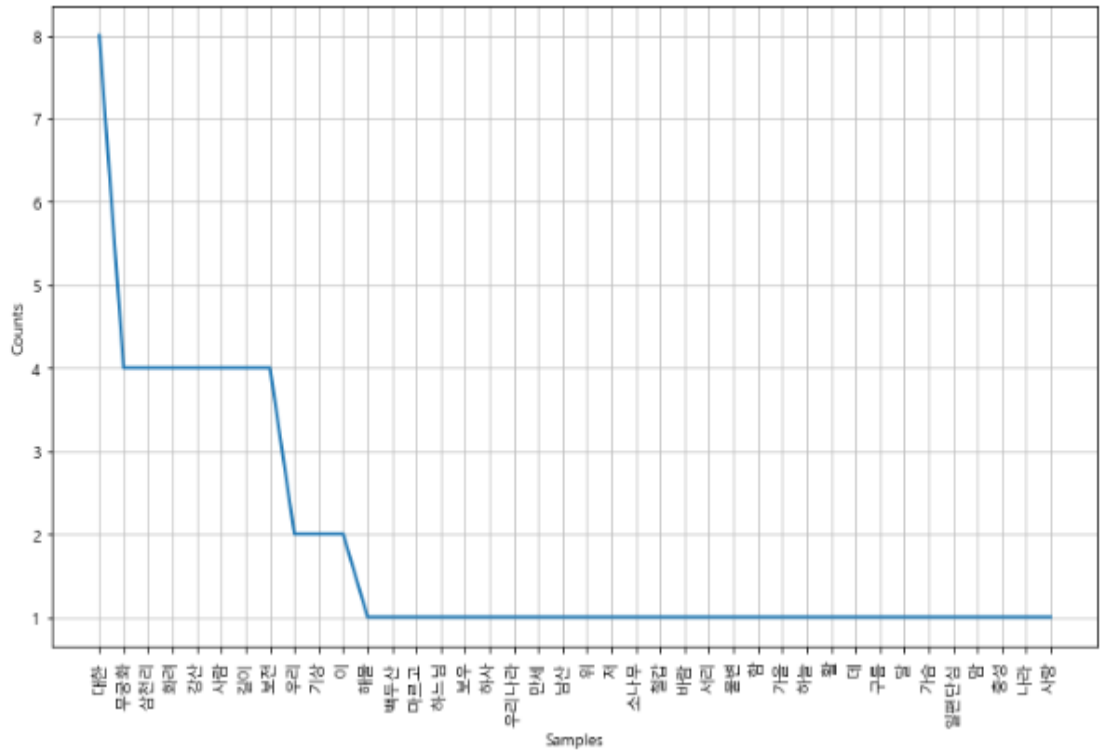
```
40
```

```
# 단어들의 빈도수 체크
ko.vocab()
```

```
FreqDist({'대한': 8, '무궁화': 4, '삼천리': 4, '화려': 4, '강산': 4, '사람': 4, '길
이': 4, '보전': 4, '우리': 2, '기상': 2, ...})
```

```
# 그래프로 확인

plt.figure(figsize=(12,7))
ko.plot()
```



불용어 처리

```
stopword = ['이','저','데']
ko = [i for i in ko if i not in stopword]
print(ko)
```

```
Out[106]: ['해물',
           '백두산',
           '마르고',
           '하느님',
           '보우',
           '하사',
           '우리나라',
           '만세',
           '무궁화',
           '삼천리',
           '화려',
           '강산',
           '대한',
           '사람',
           '대한',
           '길이',
           '보전',
           '남산',
           '위',
```

연관있는 단어들을 뽑아줌

```
ko = nltk.Text(ko)
ko.concordance('하늘')
```

```
Displaying 1 of 1 matches:
우리 기상 무궁화 삼천리 화려 강산 대한 사람 대한 길이 보전 가을 하늘 활 구름 달 우리
가슴 일편단심 무궁화 삼천리 화려 강산 대한 사람
```

```
ko_vocab = ko.vocab()  # 상위 30 개만 뽑아내기 / 리스트 모양으로 바뀜
data = ko.vocab().most_common(30)
print(data)
```

```
Out[113]: [('대한', 8),
           ('무궁화', 4),
           ('삼천리', 4),
           ('화려', 4),
           ('강산', 4),
           ('사람', 4),
           ('길이', 4),
           ('보전', 4),
           ('우리', 2),
           ('기상', 2),
           ('해물', 1),
           ('백두산', 1),
           ('마르고', 1),
           ('하느님', 1),
           ('보우', 1),
           ('하사', 1),
           ('우리나라', 1),
           ('만세', 1),
           ('남산', 1),
```

```
# wordcloud 로 만들기

wordcloud = WordCloud(font_path='C:₩windows/fonts/malgun.ttf',  ₩
                      stopwords = STOPWORDS,  ₩
                    background_color = 'white',  ₩
                  width = 1000, height = 800).generate_from_frequencies(dict(data))
plt.figure(figsize=(10,10))
plt.imshow(wordcloud)
plt.axis("off")
plt.show()
```


화려
대한
삼천리
무궁화
보전
우리
길이
강산
남산
하사
백두산
서리
바람
마르고
우리나라
구름
철갑
가을
불변